Brandon Cao
CIS 425
Spring 16’

Assignment 2

1)

(lambda (y) y) y)

2)

a) λx.(λy.xy)y

- Free variables:
  - Outside expression: y
  - Inside Expression: x
- Bound variables:
  - Outside expression: x
  - Inside expression: y

b) λk.k(λf.hf)(qf)

- Free variables:
  - Outside expression: q and f
  - Inside expression: h
- Bound variables:
  - Outside expression: k
  - Inside expression: f

3)

a) [(λy.xy)/x] (x(λx.xy))

((λy.xy)(λx.xy)) $\rightarrow$
((λy.xy)(λz.zy)) *rename x variable in second expression because the x values are different. In the end, the expression stays the same because there is no free variable x to replace.

b) [(λx.xy)/x] (λy.x(λx.x))

[(λx.xy)/x] ((λy.x)(λz.z))$\rightarrow$
[(λx.xy)/x] ((λv.(λx.xy)(λz.z))

4)

a)

λxyz.(xz)(yz)(λxy.x)(λx.x)(λx.x)→
λx.λy.λz.(xz)(yz)(λxy.x)(λx.x)(λx.x)→
(λy.λz.((λxy.x)(z))(yz)) $\underline{(\lambda x.x)(\lambda x.x)}$ → the underlined expressions are the same, so combine them into one.
(λz.((λxy.x)(z))((λx.x)z) (λx.x) →
((λxy.x) (λx.x))((λx.x) = ((λx.λy.x)(λx.x))(λx.x) →
(λy.λx.x)(λx.x) →
(λx.x)

b)

((($\underline{\lambda x.xx)}$ (λx.x))) (λx.x) →the underlined expression can be rewritten as
((λx.x)(λx.x)) (λx.x)) (λx.x)→
((λx.x)(λx.x))(λx.x)→
(λx.x) (λx.x) → Both are the same, so combine them into one.
(λx.x)

5)
a) To find the sum of the squares of the first five numbers, we would use the map function to square each number in the array and insert them into a new array. Then use reduce function to calculate the sum of the array with the squared values.

Code: reduce(function(x,y){return x+y}, map(function(x){return x^2}, [1, 2, 3, 4, 5]))

b)
To find the number of positive numbers, we first use map function to return an array where all negative values in the equal to zero. If the number at that index is positive, the value is set to one. Using the reduce function we pass in the array where the function will calculate the sum of the array.

Code:

```
1)var arr = [array of values]

2)var postiveNum = map(function(x){if(x >= 0){return 1;} else{return 0;}}, arr);
```

3)reduce(function(x,y){return x+y}, postiveNum)

c)

Using the reduce function we can flatten the arrays.

1) var z = [[1,2],[3,4],[5,6],[7,8,9]]
2) var flatArray = reduce(function(x,y){return x.concat(y);}, z);

flatArray is the concatenated form of the given array of arrays.

6)

a)
g(f) = 15

b) x = 10 & y = 7

c)
z, which equals g(f) will call the function g(h). g(h) returns h(x). x is then set to 7, so h(7) == f(7). Since the function f(y) takes in the parameter “y”, y = 7.

d)
x would equal 10 because the last time that x was set before z = g(f) is called was 10. x = 10 is at the top of the stack.

e)
g(f) = 10

f) x= 5 && y = 7

g) First, z = g(f) will call the function g(h) which returns h(x). h(x) in this case is the function f. x is then set to 7, so h(7) actually equals to f(7). The function f(y) takes in an argument y = 7 in this case. That is how y is set.

h)

x would equal to 5 because it is set in an outer block than when x is assigned to 10.

7)
λg.g(g)

(λg.g(g))(λg.g(g)) →
(λg.g(g))(λg.g(g))

We are left with the same lambda calculus expression. Therefore, the function is recursive.

8)

S_n = n+1

0 = λx.λy.y // x is presented zero times

1 = λx.λy.xy // x is presented 1 time

2 = λx.λy.x(xy) //x is presented twice

N = λx.λy.x^ny // x is presented n times

The successor function applied to the Church form of 1 would return the Church form of 0.

For this problem I will be using the variables u and v for x and y for writing the numeral form.

0 = λu.λv.v

S = λu.λx.λy.x(uxy)

S0 = S(λu.λv.v) →

[λx.λy.xy/u](λu.λx.λy.x(uxy)) →

λx.λy.x((λu.λv.v)xy) →λx.λy.x((λv.v)y) →

λx.λy.x(y) = 1

S1 = S(λu.λv.uv) →

λx.λy.x((λu.λv.uv)xy) →

λx.λy.x((λv.xv)y) →

λx.λy.x(xy) = 2

$Sn = S(\lambda u.\lambda v.u^n v) \rightarrow$

$\lambda x.\lambda y.x((\lambda u.\lambda v.u^n v)xy) \rightarrow$

$\lambda x.\lambda y.x((\lambda v.x^n v)y) \rightarrow$

$\lambda x.\lambda y.x(x^n y) = \lambda x.\lambda y.x^{n+1} y$